Quiz 1 Practice Questions

Problem 1: Count One

Consider an integer *n* stored in memory location x5000, write a program in LC-3 Assembly to count the number of 1s in its binary representation and store the result back into memory location x5000.

Example input (in R6)	Binary representation (16 bits)	Example output (in memory X5000)
0x0009	0000 0000 0000 1001	2
0x00FF	0000 0000 1111 1111	8
0x000A	0000 0000 0000 1010	2

Implement the **Brian Kernighan's Algorithm** as follows:

1 Initialize *count* = 0
2 **If** integer *n* is not zero
(a) Do bitwise & with (*n*-1) and assign the value back to *n*. That is, *n* = *n* & (*n*-1)
(b) Increment count by 1
(c) Go to step 2
3. **Else** return *count*

The code you need to finish is marked by "TODO" in the file **count_bit1.asm**. We have provided you a simple input test file input.asm. Use the following procedures to test your program:

```
~$ lc3as input.asm
~$ lc3as count_bit1.asm
~$ lc3sim
```

In lc3sim console, load the **input1.obj** to initialize the memory first and then load **count_bit1.obj** to run your program:

```
file input.obj
file count_bit1.obj
finish
```

The final result should be stored in memory location x5000.

Problem 2: Print Number in Base 7

Write a program in **base7.asm** to print a positive value stored in R3 in base 7 format.

Algorithm
1) Divide value stored in R3 by 7. Store quotient in R3 and push remainder to stack.
2) If quotient (value in R3) is not 0, go to step 1
3) Pop values off the stack one at a time till the stack is empty. Add ASCII offset for '0' and print to screen. You do not need to print the 'x' in front.

You can use any TRAP you find useful. The DIV, PUSH and POP subroutines are given to you. PUSH and POP subroutines are the same as the ones given in lab and MP. Stack starts at x4000 and ends at x3FF0, which means the first available spot on the stack is at x4000 and the last available spot at x3FF0. You are not required to use subroutines in this problem, but you may write subroutines if you like.

Testing:

Assembly your code using the command:

~$ lc3as base7.asm

You may test your code using the following command and set R3 to 9:

~$ lc3sim base7.obj

register R3 9

finish

Problem 3: Fibonacci Sequence

For this problem you will write a subroutine in the file **fibonacci.asm** that calculates F_k, the kth Fibonacci number. The subroutine should take input k from R1, and save F_k in R6.
Fibonacci numbers are numbers in the integer sequence 1,1,2,3,5,8,13,… By definition, $F_1 = 1$ and $F_2 = 1$. Each subsequent Fibonacci number is equal to the sum of the two previous numbers.

$$F_i = \begin{cases} 1 & , i = 1 \\ 1 & , i = 2 \\ F_{i-1} + F_{i-2} & , i \geq 3 \end{cases}$$

A table of relevant Fibonacci numbers appears at the end of the problem.

INPUTS

K : A positive integer number stored in **R1**. Your program does not need to handle invalid inputs.

OUTPUTS

$\boldsymbol{F_k}$: The kth Fibonacci number. **The output should be saved in R6**. If F_k is too large to be represented with 16 bit two's complement, the program should output **–1**.

Algorithm

An algorithm to calculate F_k iteratively is shown below in pseudocode. You are **not** required to implement your subroutine this way. Please note that this algorithm is not recursive (like the one discussed in class). It builds the Fibonacci sequence from the lower values up to the higher values.

```
If k<= 2
        Fk = 1
If k => 3
        F_low = 1
        F_high = 1
        For i = 3 to k:
                F_sum = F_low + F_high
                F_low = F_high
                F_high = F_sum
        Fk = F_sum
```

The first 26 Fibonacci numbers:
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368